Listing of Claims:

1-41. (Canceled)

42. (Previously presented) An apparatus, comprising:

a machine-readable storage medium; and

instructions in the machine-readable storage medium, wherein the instructions, when executed by a data processing system, implement an interpreter to perform operations comprising:

receiving a series of source code instructions expressed in a programming language, wherein at least one of the source code instructions comprises a command and an argument;

in response to receiving a source code instruction having a command and an argument, building at least part of a stack-based execution stream; and

after building at least part of the stack-based execution stream, executing the stack-based execution stream;

wherein the operation of building at least part of a stack-based execution stream comprises:

storing the instruction's argument on a stack;

determining an address for an object code routine corresponding to the instruction's command; and

storing the address for said object code routine on the stack.

43. (Previously presented) An apparatus according to claim 42, wherein the operation of executing the stack-based execution stream comprises using the address of the object code routine from the stack to call said object code routine.

44. (Previously presented) An apparatus according to claim 42, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions; and

the operation of building at least part of a stack-based execution stream comprises:

pushing the instruction's argument onto the stack from the parser routine; and

pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine.

45. (Previously presented) An apparatus according to claim 42, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions;

the operation of building at least part of a stack-based execution stream comprises:

(a) pushing the instruction's argument onto the stack from the parser routine; and

(b) pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine; and

the operation of executing the stack-based execution stream comprises:

(a) calling an interpreter routine from the primary routine, wherein the interpreter routine uses the address of the object code routine from the stack to call said object code routine;

(b) retrieving the argument from the stack, wherein said retrieval is performed by the object code routine;

(c) using the argument when executing the object code routine; and

(d) storing a result of the object code routine on the stack.

46. (Previously presented) An apparatus according to claim 42, wherein the operation of executing the stack-based execution stream comprises:

retrieving the argument from the stack; and

using the argument when executing the object code routine.

47. (Previously presented) An apparatus according to claim 42, wherein the operation of executing the stack-based execution stream comprises:

retrieving the argument from the stack;

using the argument when executing the object code routine; and

storing a result of the object code routine on the stack.

48. (Previously presented) An apparatus according ot claim 42, wherein the operation of executing the stack-based execution stream comprises:

recursively executing the object code routine.

49. (Previously presented) A method for executing source code instructions expressed in a programming language, the method comprising:

receiving a series of source code instructions expressed in a programming language, wherein at least one of the source code instructions comprises a command and an argument;

in response to receiving a source code instruction having a command and an argument, building at least part of a stack-based execution stream; and

after building at least part of the stack-based execution stream, executing the stack-based execution stream;

wherein the operation of building at least part of a stack-based execution stream comprises:

storing the instruction's argument on a stack;

determining an address for an object code routine corresponding to the instruction's command; and

storing the address for said object code routine on the stack.

50. (Previously presented) A method according to claim 49, wherein the operation of executing the stack-based execution stream comprises using the address of the object code routine from the stack to call said object code routine.

51. (Previously presented) A method according to claim 49, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions; and

the operation of building at least part of a stack-based execution stream comprises:

pushing the instruction's argument onto the stack from the parser routine; and

pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine.

52. (Previously presented) A method according to claim 49, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions;

the operation of building at least part of a stack-based execution stream comprises:

(a) pushing the instruction's argument onto the stack from the parser routine; and

(b) pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine; and

the operation of executing the stack-based execution stream comprises:

(a) calling an interpreter routine from the primary routine, wherein the interpreter routine uses the address of the object code routine from the stack to call said object code routine;

(b) retrieving the argument from the stack, wherein said retrieval is performed by the object code routine;

(c) using the argument when executing the object code routine; and

(d) storing a result of the object code routine on the stack.

53. (Previously presented) A method according to claim 49, wherein the operation of executing the stack-based execution stream comprises:

retrieving the argument from the stack; and

using the argument when executing the object code routine.

54. (Previously presented) A method according to claim 49, wherein the operation of executing the stack-based execution stream comprises:

retrieving the argument from the stack;

using the argument when executing the object code routine; and

storing a result of the object code routine on the stack.

55. (Previously presented) A method according to claim 49, wherein the operation of executing the stack-based execution stream comprises:

recursively executing the object code routine.

56. (Previously presented) A data processing system, comprising:

a processor;

a machine-readable storage medium responsive to the processor; and

instructions in the machine-readable storage medium, wherein the instructions, when executed by the processor, implement an interpreter to perform operations comprising:

receiving a series of source code instructions expressed in a programming language, wherein at least one of the source code instructions comprises a command and an argument;

in response to receiving a source code instruction having a command and an argument, building at least part of a stack-based execution stream; and

after building at least part of the stack-based execution stream, executing the stack-based execution stream;

wherein the operation of building at least part of a stack-based execution stream comprises:

storing the instruction's argument on a stack;

determining an address for an object code routine corresponding to the instruction's command; and

storing the address for said object code routine on the stack.

57. (Previously presented) A data processing system according to claim 56, wherein the operation of executing the stack-based execution stream comprises using the address of the object code routine from the stack to call said object code routine.

58. (Previously presented) A data processing system according to claim 56, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions; and

the operation of building at least part of a stack-based execution stream comprises:

pushing the instruction's argument onto the stack from the parser routine; and

pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine.

59. (Previously presented) A data processing system according to claim 56, wherein:

the operation of receiving a series of source code instructions expressed in a programming language comprises repeatedly calling a parser routine from a primary routine for interpretively executing source code instructions;

the operation of building at least part of a stack-based execution stream comprises:

(a) pushing the instruction's argument onto the stack from the parser routine; and

(b) pushing the address for the object code routine corresponding to the instruction's command onto the stack from the parser routine; and

the operation of executing the stack-based execution stream comprises:

(a) calling an interpreter routine from the primary routine, wherein the interpreter routine uses the address of the object code routine from the stack to call said object code routine;

(b) retrieving the argument from the stack, wherein said retrieval is performed by the object code routine;

(c) using the argument when executing the object code routine; and

(d) storing a result of the object code routine on the stack.

60. (Previously presented) A data processing system according to claim 56, wherein the machine-readable medium comprises at least one medium from the group consisting of:

random access memory;

non-volatile memory; and

disk based storage.